Problem sheet 1

Thomas E. Woolley

Last edited on:

November 4, 2020

1 Early warfare

Originally[1], warfare was conducted through hand to hand combat. Thus, a single person had the ability to despatch only one other person at a time. Namely, the combat was *one on one* until one person was unable to fight any longer. Consider two opposing armies of size $A(t)$ and $B(t)$ and suppose $A(0) = A_0$, $B(0) = B_0$, where $A_0 > B_0$.

1. Suppose the armies are equally matched such that when members of the A and B armies meet they are both equally like to win the fight. Further, assume that the interaction rate constant between the two armies is r, write the above combat interaction down as two interaction equations.
2. Show that the Law of Mass Action can be used to convert the early warfare interactions into the following ODEs,
$$\dot{A} = -rAB, \quad A(0) = A_0, \tag{1}$$
$$\dot{B} = -rAB, \quad B(0) = B_0. \tag{2}$$
3. Derive the steady states of equations (1) and (2). What happens when you try to characterise their stability?
4. Show that
$$A(t) = B(t) + \text{Constant}, \tag{3}$$
where you should define the constant explicitly.
5. In question 3 you should have found that there was an infinite number of steady states. Show how equation (3) collapses all possibilities to just one.
6. Construct the (A, B) phase plane and use it to characterise the steady states stability.
7. Show that equations (1) and (2) can be rewritten as
$$\dot{A} = -rA(A - A_0 + B_0), \tag{4}$$
$$\dot{B} = -rB(B + A_0 - B_0). \tag{5}$$
8. Derive the steady states of equations (4) and (5).
9. Considering equations (4) and (5) as separate scalar ODEs characterise the stability of the previously derived steady states analytically and graphically through constructing the phase planes $(A, \dot{A})$ and $(B, \dot{B})$.
10. Consider equations (4) and (5) as a system and characterise the stability of the steady states algebraically and graphically by plotting the (A, B) phase plane.
11. Do the results from questions (9) and (10) correspond to the results from question 6?
12. Solve equations (4) and (5) explicitly and sketch the resulting $(t, A(t))$ and $(t, B(t))$ curves on the same axis.
13. You have now derived the same result in three different ways. Without referring to the mathematics, interpret your result.

[1]This question is an adapted version of Lanchester's Laws. Have a look into how modern warfare differs from early warfare.

2 Non-dimensionalisation and interpretation

Consider the following set of interaction equations for two animal populations N and P.

$$\frac{\mathrm{d}N}{\mathrm{d}t} = rN\left(1-\frac{N}{K}\right) - bNP, \tag{6}$$

$$\frac{\mathrm{d}P}{\mathrm{d}t} = ebNP - mP. \tag{7}$$

where b, e, K, r and m are positive constants and the initial conditions are N_0 and P_0, respectively.

1. What type of interaction is occurring between N and P?
2. Write down a set of interaction equations that could give rise to equations (6) and (7).
3. Non-dimensionalise equations (6) and (7) to get the form

$$\frac{\mathrm{d}u}{\mathrm{d}\tau} = c_1 u(1-u) - uv, \tag{8}$$

$$\frac{\mathrm{d}v}{\mathrm{d}\tau} = c_2 uv - v, \tag{9}$$

   where c_1 and c_2 should be defined explicitly in terms of K, b, e, r and m.

4. Provide an interpretation of c_1 and c_2.
5. Suppose the predator is a pest and we want to wipe the predator out, whilst maintaining a positive prey population (*e.g.* consider a fox and hen scenario). What strategy best achieves this, in terms of altering the sizes of c_1 and c_2? Explicitly, is it best to:
   - increase c_1?
   - increase c_2?
   - increase both?
   - decrease c_1?
   - decrease c_2?
   - decrease both?
   - increase c_1 and decrease c_2?
   - decrease c_1 and increase c_2?

   Explain your choice.

6. Having chosen a strategy consider how best to enact the changes of c_1 and/or c_2 in terms of altering b, e, K, r and m. What are the pros and cons? What would your advice be of best strategy?

3 Using `ode45`

As in MA0232 I will be offering you the chance to have a go at understanding Matlab. It is free to download, all you need is follow this (clickable) link `https://uk.mathworks.com/academia/tah-support-program/eligibility.html` and sign up using your Cardiff email address.

Throughout this course we will be simulating ODEs of many types and, thus, it is good to be able to check your answers. The more you simulate, the better your intuition. In order to simulate ODEs we are going to use Matlab's inbuilt command `ode45`. Matlab has several ODE solvers, but `ode45` is the most general "catch all" code. Essentially, you try it first and if it does not work you try something more specific. However, for our purposes it will be adequate.

For a high level description of `ode45` type `help ode45`, or `doc ode45` into Matlab and it will bring up a brief description and a full description, respectively. Here, I will provide a brief overview of a general code that will work on most systems throughout this course.

The code below should be available as `General_ode45_code.m` and it solves problem 1. Either download it directly, or copy and paste it into Matlab. The green text are comments to help you understand what is going on. Essentially, for different ODE systems you should only really have to change the description in lines 49-50. Note you may also need to change the length of time over which the equations are solved (`Tend`).

```
%% Ensure we start from a blank slate
clear all
close all
clc

%% Initialise variables
fs=15; % Set fontsize

Tend=0.3; % Set final time
N=100; % Set number of steps
t=linspace(0,Tend,N); % Set number of time points at which the solution is to be evaluated
IC=[200;
    190]; %Initial conditions as a column vector

%% Solve ODE

% Each component is a row in the matrix y.
[t,y] = ode45(@ODE_function,t,IC);

%% Plot Solution

%%% (time,solution plots)
hold on % Allows us to plot multiple graphs at once
plot(t,y(:,1)) % Plots the first solution
plot(t,y(:,2)) % Plots the second solution
legend('u','v') % Adds a legend and defines the lines for clarity

% Label axes for clarity
xlabel('t')
ylabel('Solutions')

set(gca,'fontsize',fs) % Changes the axis fontsize

%%% Phasespace plots
figure % Starts a new figure without overwriting the previous one
plot(y(:,1),y(:,2))
xlabel('u')
ylabel('v')
set(gca,'fontsize',fs)




%% ODE description
function dydt=ODE_function(t,y)
r=1; %Interaction parameter

% Defining the ODEs as a column vector. Each row is one of the ODEs.
dydt=[-r*y(1)*y(2) %Line 49
      -r*y(1)*y(2)]; %Line 50

end
```

Once you have had a play with the above code try and adapt it to solve equations (8) and (9).

Exam Revision

Problem sheets 3-5 of course MA0232 contain many examples and solutions that will provide adequate practice at: deriving steady states, non-dimensionalisation, characterising stability and sketching graphs.